A New Paradigm for Internet Browser Design

or

An Alternative Method for Building an Internet Browser

BACKGROUND OF THE INVENTION

The present invention discloses a method for displaying World Wide Web content or any content that might be viewed by an HTML-based browser. This information access is not limited to HTML data, but it could be any data being accessed including Intranet data, HDML (handheld device markup language) or WML (wireless markup language) based information. This method leverage existing standards and technology in a way that could improve all browsing methods in general and certainly solves some major problems with wireless handheld devices. In the preferred embodiment the information is most likely an Internet or Intranet world-wide web (WWW) pages, with HTML, HDML or WML content, that is accessed by the user. In particular, the invention discloses the use of a Virtual Machine (VM) as the browser itself. Using a VM as the browser itself has two immediate impacts on the viewing paradigm these include: delivering programs as content to the viewer and converting existing HTML, HDML and WML content into programs for the viewer. In the preferred embodiment the VM of choice would be Java VM running the new Micro Edition of the Java definition. The impact of this invention is dramatic, and effectively means that large HTML browsers can be reduced to a small footprint VM for interpreting programs. For small devices, such as cell phone, wireless handheld devices and palm-top computers, this small footprint is essential for creating a solution that can run in the limited space available.

For those skilled in the art of Internet Browsing will appreciate the large number of problems that are growing and are remaining unsolved. These problems include:

(a) the growing incompatibility between content sent and rendered by browsers in the marketplace like Netscape and Internet Explorer,

(b) the growing divisions between 'scripting' languages used by browsers in the Internet, for example Javascript used by Netscape and Visual Basic Script used by Internet Explorer,

(c) current browsers must not only render HTML but also contain a Java compiler for executing Java programs, this leads to very large and complex browser programs that can work only on the largest and fastest of desktop systems,

(d) the growing size and complexity of web pages and their formats to address the ever increasing need for animation and active web page content, this is placing major strain on the Internet infrastructure and is impossible to support on wireless devices with very limited bandwidth,

(e) finally the problem of ever changing browsers, versions, updates and deployment to the marketplace.

As a result of all the aforementioned problems and complexity, there remains a further need for a system that provides the development community a better method for viewing content to their handheld device, cell phone and desktop systems. The present invention solves addresses most of the problems listed above in unique ways. The improvements and solutions to these problems include:

(a) By using a language all Virtual Machine (VM) interpreters follow the same standard language. Languages like 'C' or Java are now considered a mature language like the 'C' language and changes are rare and far between.

(b) The use of scripting languages within HTML content is a stop-gap method designed mostly for parameter verification, animation support and supporting active web page content. The use of a full language provides much more control and flexibility then a scripted, interpreted set of instructions.

(c) The current use of the Java language with exiting browsers provides limited program execution within a browser program designed for HTML rendering. Supporting Java within the browser is resulting in very complex and large browsers that dramatically limits where the browser can be used. Eliminating the HTML component of the browser, and making it only a VM means the size and complexity are dramatically reduced. This also means that a developer with lots of space could make a VM

program whose job it is to interpret HTML tags and syntax. Multiple programs can support inter-task communications, parameter passing and a wide variety of programmatic methods that those skilled in the art can appreciate are not there today.

(d) By using a programming language instead of a mark up language the developer will be able to do infinitely more complex tasks in less space. As the tasks of animation, advertising, as active web page development continue into the program development space, the use of a real programming language make providing these features easier and easier. Currently companies are stretching HTML and scripting to their limit to "keep the eye of the consumer" and it is putting heavy demands on the Internet capacity and backbone.

(e) Another exceptional feature of using Java to present information to the user is that the program interpreter running as the viewer rarely needs to change. The sophistication of the program being delivered can increase and get more complex but the interpreter can stay the same for a very long time and offer continued increasingly complex solutions. This can especially help solve additional problems in the area of browser size. To execute some of these 'complex' mark up pages the browsers are growing into monolithic sized programs. These large browsers are incapable of running on small device, palm-tops computer, handheld phones and wireless devices. By running a program interpreter on the device the programs size can be kept thin and efficient as necessary for the environment.

(f) By using a pure Virtual Machine as the 'base-line' of the product it is amazingly easy to upgrade and add new functionality. In today's market to get an update to a browser requires a 56K baud modem or connection to the Internet and about 30 minutes of download time. This is mostly because it is a one-size serves all and every component and piece of HTML functionality must be in every browser shipped. By offering a flexible method for delivering the browser as a Java application, the browser software can start out as a small subset of HTML and grow as the need and technology evolves. This kind of staged evolution of browser technology is exactly what small devices and wireless devices need today for the support of Internet information.

The conclusion to all to the advantages just listed is that the ability to offer a browser solution to a wider number of devices is now possible using an industry standard technique. Those skilled in the art know that the problems facing wireless devices wanting to access Internet-based information are enormous. By using this invention the wireless industry can move to converting HTML, WML and other formats to Java programs and then sending bandwidth friendly Java programs to be executed on the handheld device. By using a programming language like Java the information creator has the ability to use native language methods to solve complex problems. No longer does an HTML implementor have to stretch the formatting language to do complex presentations that it was never designed to achieve.

In the preferred embodiment the language being used is Java and a Java virtual machine is being used to interpreter and execute the program. The program is being executed on a wireless handheld device with limited memory, screen size and CPU capabilities. In this environment the throughput expectations are about 2400 to 9600 baud with a very bursty delivery depending on coverage and roaming parameters. Additional to the claim it is possible that a proxy server or gateway, acting on behalf of the handheld wireless user, is able to fetch existing HTML, HDML and WML Internet and Intranet content and convert it to Java programs for execution. When a proxy is being used for translating HTML, WML or HDML into Java programs, it is able to do this in real-time to satisfy information requests from the user. This translation process allows for legacy systems to be supported with the new advanced methods to support wireless devices. When a gateway is present in the solution it can be used to tokenize the Java programs and further reduce their over-the-air payload size and thus increase effective speed in reaching the handheld wireless device.

This invention provides new possibilities to all Internet viewing points, from cell phone viewers, handheld device and regular desktop systems. The ability to improve the sophistication of the viewer without changing the viewer is a huge deployment advantage and places the host system providing the information in charge with how complicated each program/page is.

SUMMARY OF THE INVENTION

The present invention overcomes the problems previously noted in the area of Internet browsers, and solves the need of transferring sophisticated programs and web content to small handheld devices. In solving the problem of browser complexity and deployment, the invention proposes using a Virtual Machine (VM) in place of a massive page-rendering engine. This allows the oversized browser to be broken into sub-components. An information viewer could first download a small HTML page-rendering program using HTTP and then start viewing HTML pages on the Internet using the downloaded program. Additionally a small cell phone user could first download a small Wireless Markup Language (WML) page-rendering engine using HTTP, and then start to access WML web sites offering this specialized information content for handheld devices. The invention solves problems with every growing web site page content by using a programmatic language to replace a page-rendering language. Problems like scripting, animation, special window controls, and pop-up advertising is being performed with a combination of HTML, Java-script and Java all of which is leading to a large complex page that is clogging the Internet capacity limits. For those skilled in the art it is clear that by using a pure language the developer can focus on getting the best solution to the problem using a native language calls and commands. When converting HTML or WML to this VM language the content can be reduced to its smallest size using a tokenization technique commonly practiced in the industry.

The preferred client for viewing Internet information would be a mobile device on a wireless digital data network. Once of these networks is the Mobitex Radio Network ("Mobitex"), which has been developed by Eritel and Ericcson of Sweden, and is operated by BellSouth Wireless Data in the United States. Another network is the DataTAC Radio Network ("DataTAC"), which has been developed by Motorola/IBM and is operated by American Mobile Satellite Corporation (AMSC) in the United States and Bell Mobility in Canada.

According to the method of the present invention any information source accessed via a TCP/IP network, using a standard information retrieval protocol like Hyper-text Transfer Protocol (HTTP) will be able to retrieve pure software programs to be executed, in place of traditional marked-up pages that must be rendered. The traditional Internet browser or viewer is replaced with a virtual machine interpreter that is capable of dynamic program download and execution providing advanced program behaviours and controls. This is done by (i) defining a language to be used in place of HTML, HDML or WML in this patent the preferred embodiment is to use Java; (ii) making a request for information from a wireless handheld device, cell phone, or traditional desktop from the program interpreter using a traditional universal resource identifier/locator (URI or URL); (iii) sending a program to display the information in the program interpreter using the language that was chosen for the purpose. The information source can either be a traditional web server or a gateway machine making the request on behalf of a handheld wireless device or cell phone device. The program interpreter on the handheld device has no direct HTML, WML or HDML components, only the VM that is used to execute programs that are downloaded. It is therefore possible to download a personality, like a WML formatter into the device and start viewing information in a more conventional way.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention creates a new standard for browsing information and working with worldwide web (www) sites. The invention defines the ability to use a programmatic language to present information to users and that the browser is entirely composed of a Virtual Machine (VM) program interpreter. The following diagrams will help illustrate the invention, and they include:

Figure 1 is typical overview of the system where the invention could be used. It shows both a direct TCP/IP link between the information source 100 and the Virtual Machine and File Explorer 500, and a network connection. In the preferred embodiment the network connection 300 would be over a wireless data network and the device side would be a small handheld wireless device like a cell phone or a pager.

Figure 2 shows a more detailed overview of the optional gateway component 200, first shown in figure 1. The optional gateway component can be used make the invention inter-connection with legacy systems and legacy information sources like HTML and WML. In this way the invention can be made backward compatible so that original information sources can still be viewed if necessary from a device running the Virtual Machine 500.

Figure 3 shown a more detailed overview of the Virtual Machine (VM) and File Explorer 500, first shown in figure 1. These components are central to the inventions operation and usage.

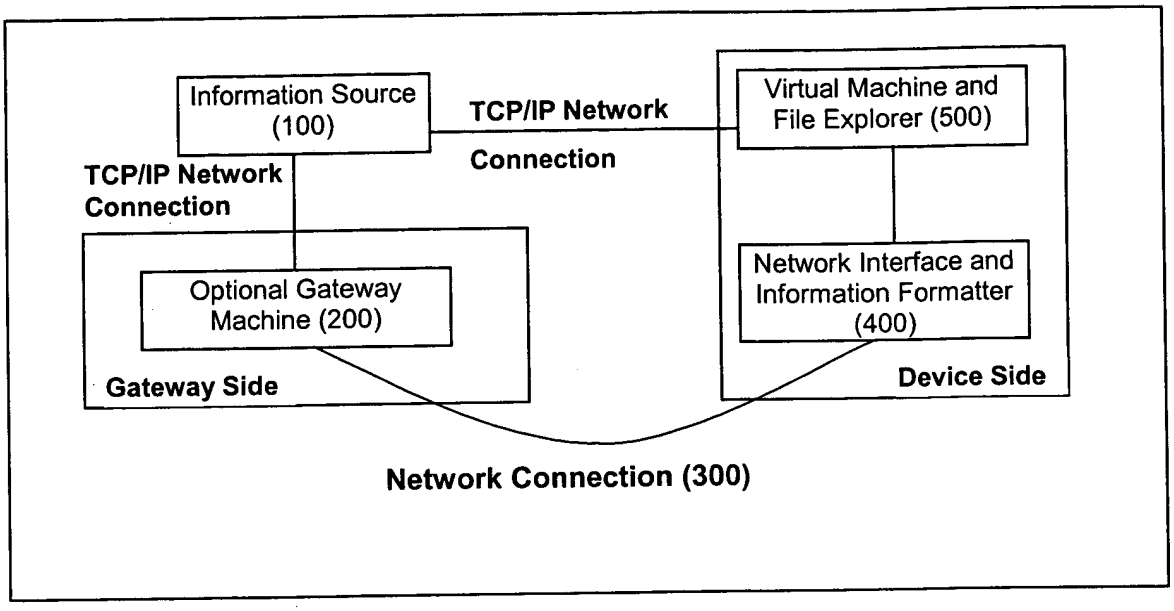

Figure 1 –Overview of System

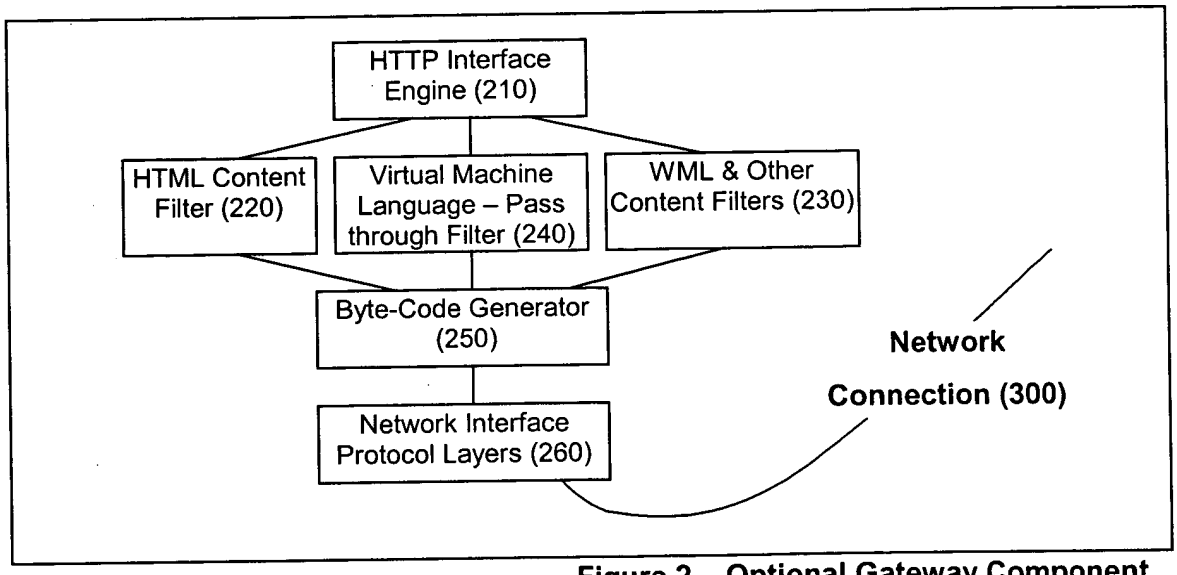

Figure 2 – Optional Gateway Component

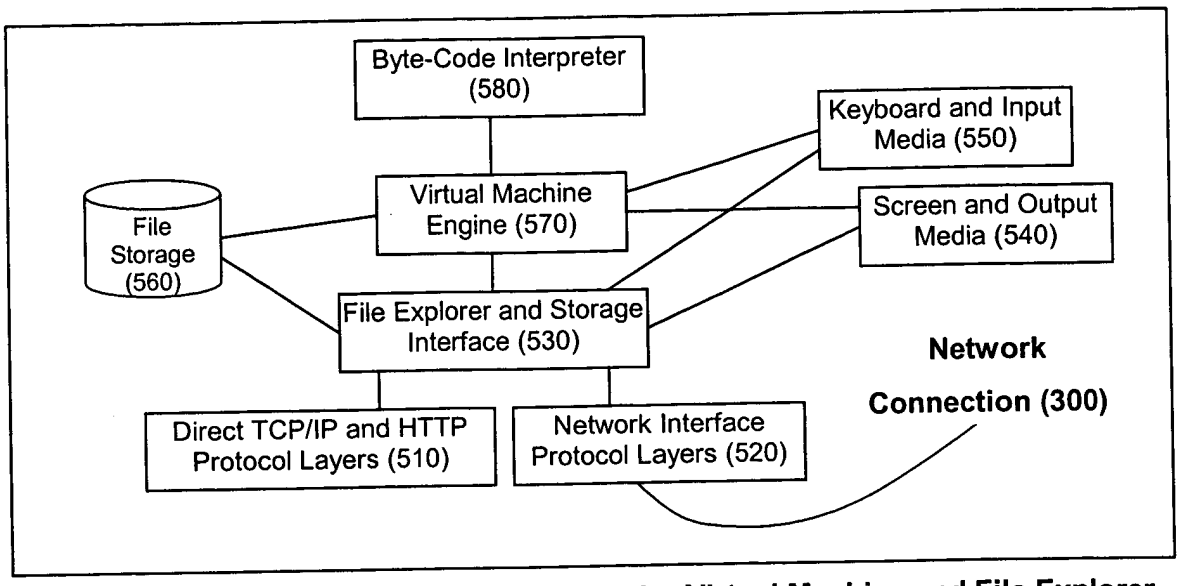

Figure 3 – Virtual Machine and File Explorer

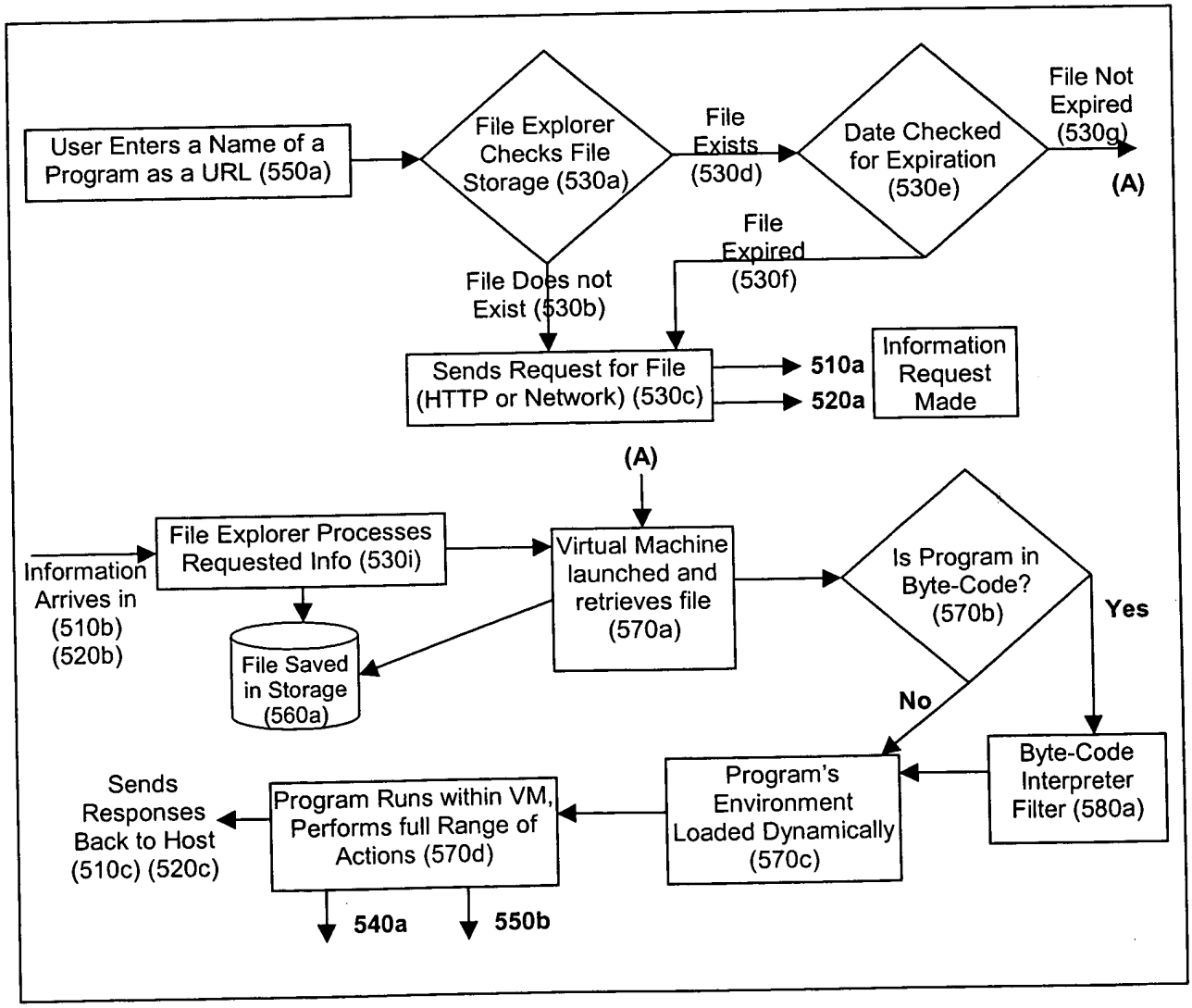

Figure 4 – Logic for Fetching and Executing a Program

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring now to the drawings, Figure 1 represents an overview of the environment where the invention is most likely to be used. The first element of the system is that an information source 100 is required, as a repository of information and data desired by individuals who are in remote proximity to that information. Typically this remote access to an information source is achieved by using the TCP/IP protocol to reach the data. Those skilled in the art will appreciate that TCP/IP is not enough to access data but a protocol like HTTP used in conjunction with TCP/IP serves as a method for retrieving data. To access the information, either a TCP/IP connection from a Gateway 200 or a direct TCP/IP connection from the Virtual Machine and File Explorer 500 is required. In the case where the Virtual Machine and File Explorer 500 understands TCP/IP and the HTTP protocol and the information source is delivering native VM programs this is a very viable connection alternative.

The optional Gateway 200 is used as a bridge between the Internet and a given incompatible Network Connection 300 like Mobitex or Datatac mentioned earlier. In these cased the Gateway 200 performs the necessary connection and protocol requirements for both the TCP/IP network and the incompatible network 300. In the preferred embodiment this network connection 300 is a wireless data network, with limited bandwidth and high data delivery latency making TCP/IP fail when used directly.

Across the network connection 300 is the network interface and information formatter 400 on the device side. This is an optional component that is necessary when the viewing device is in a specialized network, like a wireless data network. In these circumstances the device requires a network interface 400 to communicate to the network for the exchange of information. When this component is not required could be in a situation when the device can communicate natively over TCP/IP either over a traditional network, a serial connection or a Local Area Network (LAN) connection.

Finally there is the virtual machine and file explorer 500 component. This last component allows the user to specify an address to retrieve their information, either a Universal Resource Locator (URL), a filename or some other identifier used to target one

given piece of information. Those skilled in the art would understand that in the Internet the URL has become the key addressing mechanism to locate information sources within a given Internet domain. Once the user has identified and has retrieved a large number of VM programs they might want to view them and review what they have cached on the local device. This is also sometimes known as the history list or the favourities list and is simply a file viewer that lists the files in a sorted order, like chronological. This component is necessary so that the VM Programs can be requested by the user and re-executed later if necessary. In the case of browsing worldwide web sites the user might to see where they have been, recall cached information or delete previous references and clean up local storage.

Moving on to figure 2 this figure gives a more detailed view of the optional Gateway 200 components. The first component the HTTP Interface Engine 210 acts on behalf of the device and interfaces to the HTTP component of the information source. By using the HTTP protocol, which is standard in the Internet community the gateway is able to reach the widest possible sources of information, anything from web sites to Intranet sites and even proprietary database sources now support TCP/IP and HTTP connections.

Once the information has been fetched it is routed to a content filter for processing. The most common content filter is the HTML Content Filter 220 that converts HTML from all web sites into VM programs. This filter turns HTML commands, Java script, VB Script and other content into a standard VM program which will be turned into byte code and sent to the device for execution. On the simply side of this filter the gateway will turn standard text into something similar to print statements, which offers limited advantage. However on the complex side, where HTML is being stretched for solving complex user input decisions and animation or when using lots of Java or Visual Basic scripting content, the filter will be a great efficiency and space saver. Another common filters would probably include a Wireless Markup Language (WML) filter 230, which would also produce VM programs. Additional filters that could be created would be Extensible Markup Language (XML), and Handheld Device Markup Language (HDML). Those skilled in the art would appreciate the addition of any filter is simply an exercise in parser creation.

Another interesting aspect of the optional gateway is the ability to have a Virtual Machine 'pass through' filter 240 that would allow for information sources 100 to send native VM programs directly to the device for execution. This would be commonly used in new information sources and hosts where the developer wanted the best and fastest method for working with a specific device or set of devices.

All of these filters are then feed into an optional Byte-Code Generator 250 that accepts the VM program as input and turns the larger user-friendly English words into byte code representations. Programmatic languages use words like if, for, while, include, boolean, int, byte, char and return to create a logic flow and syntax; these words can be translated into numbers like 1, 2, 3, etc for a major reduction in size. This is an optional component because it's main goal is to reduce the size of the program that might be sent to the device. If the network didn't need the size reduction or if the device contained the ability to understand the full English words then this generator 250 could be excluded from the gateway.

Finally to connect the gateway 200 to a network there are required network and protocol layers 260 required for communication to the end-user devices for execution the VM programs to display the information. This final step might include breaking up and packetizing the information for transmission over the network, the additional of a network header for routing across the network and a verification or transport layer to ensure all pieces of the information arrived intact to the remote device.

Moving now to figure 3 this provides a detailed view of the Virtual Machine and File Explorer environment 500. Although what is being claimed is the concept of using a virtual machine programmatic language for rendering content, the file explorer is a requirement for picking and selecting the programs to be executed. The first component of this diagram is the Direct TCP/IP and HTTP Protocol Layers 510, which are needed when a direct connection is established to the information source 100. In this situation the information source 100 is delivering native VM programs that can be executed directly without the need for any gateway 200 filtering. The goal of this component, much like a traditional browser is to establish a TCP/IP connection and fetch the requested information using the HTTP protocol over TCP/IP.

When a direct TCP/IP connection with HTTP is not used a network connection is required and the Network Interface and Protocol Layers 520 component is used to communicate with the gateway's 500 network interface and protocol layers 260. As with network interfaces there are requirements for packetizing the data, packet transmission and reception, packet verification and dealing with network protocols.

Both the TCP/IP with HTTP component 510 and the Network Interface component 520 work directly with the File Explorer and Storage Interface 530 component. This component provides the initial User Interface (UI) to the user to allow them to specify what information they want and where. This is similar to a File Explorer product that one skilled in the art might see on any computer system. It lists any existing programs (VM programs) that have been previously retrieved and perhaps gives an indication of whether they have been viewed before (read or unread flag indicator). To interface to the user the File Explorer and Storage Interface 530 component uses the Screen and Output Media 540 component and the Keyboard and Input Media 550 components. For visual output the Screen and Output Media 540 is used for display to the user the existing state of the virtual machine environment. This might include file listings of programs on the machine and a section of specifying a new information name, like a Universal Resource Locator (URL), for retrieving another piece of information. Those skilled in the art will appreciate that the Output Media 540 could range very dramatically depending on the device being used. For the Keyboard and Input Media 550 component this allows the user to maneuver through the listing of programs and information sources and make selections. This input could be roller wheels, mouse input, keyboard keys and other devices not listed here but that are common in the field.

Once new information is received it is stored in the File Storage 560 area by the File Explorer 530 component. The file storage 560 component could be composed simply of RAM storage, but is more likely long-term storage made up of hard disk space, flash memory or other storage media. Those skilled in the art will understand that the storage media can vary from system to system, but the goal of having the ability to store the requested programs is important to the caching and execution model of any information viewing system.

After retrieval has taken place by the File Explorer and Storage Interface 530 component, it will launch the Virtual Machine Engine 570 with an indicator of which program to execute. The Virtual Machine Engine 570 could then either be designed to receive the program directly from the File Explorer 530 or go to the File System 560 to retrieve the program to be executed. At this point the Virtual Machine Engine 570 could either run the program, if it is in native form, or send the program to the Byte-Code Interpreter 580 for translation into native VM program. As the VM executes the program it uses the Screen Display and Output Media 540 and the Keyboard and Input Media 550 components to produce an effect on the device being manipulated. The end result is a program that can display a full range of complex actions, movements, form presentations and behaviours and accept a very complex number of keyboard inputs, mouse movements, roller wheel movements, bar code reader input and a many other possibilities that those skilled in the art can appreciate.

Moving to figure 4 this is an illustration of the logic used to determine when to fetch a given program and how to execute the program once it has been fetched. This logic follows closely the average browser's decision path that is currently available in the industry. The main difference is that once the program has been received it is given a virtual environment in which to run in and is then able to perform native calls within the device. Those skilled in the art will appreciate the range of things that can be done programmatically will dramatically extend what information loaded over a network can do once executed. In figure 4 the first step is the user's decision to enter the name of a URL or identification string (550a) on the keyboard of the device. This information is given directly to the File Explorer and storage component 530 who checks the file system 530a for the presence of the file already cached. If the file does not exist 530b then the File Explorer sends a request for the file 530c either using the HTTP method 510 or via the network interface 520.

Having described in detail the preferred embodiment of the present invention, including its preferred modes of operation and redirection, it is to be understood that this operation could be carried out with different elements and steps. This preferred embodiment is present only by way of example and is not meant to limit the scope of the present invention which is defined by the following claims.

What is claimed:

1. A method of displaying Internet information comprising the steps of:

   preparing the information to be retrieved into the form of a program,

   having a user make a request that results in the fetching the program,

   giving the fetched program to a virtual machine language interpreter for execution

   executing the fetched program as a response to the request for information.

2. The method of claim 1, wherein the device being used is a handheld device with a limited viewing area.
3. The method of claim 1, wherein the device is a wireless device.
4. The method of claim 1, wherein the data network is a bandwidth limited wireless data network.
5. The method of claim 1, wherein the information is not requested but is pushed to the device spontaneously through a previously established set of criteria.

6. A system to display Internet-based information using a viewing device with a keyboard, screen and network connections that contains the following components:

   programs are received over the network connection and are saved into storage by a file manager component,

   the file manager component then lets the a virtual machine interpreter to execute the program that it extracts from the file manager component..... etc, etc.